(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 586 768 A1

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 92480130.1

(51) Int. Cl.5: G06F 9/46, H04L 29/06

(22) Date of filing: 11.09.92

(43) Date of publication of application:
16.03.94 Bulletin 94/11

(84) Designated Contracting States:
DE FR GB

(71) Applicant: International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Basso, Claude, La Lanterne Des Anges-A3
252 Avenue de la Lanterne
F-06200 Nice(FR)
Inventor: Calvignac, Jean
187 Chemin les Vallières
F-06610 La Gaude(FR)
Inventor: Pham, Tan Thanh
Résidence "Les Bastides",
120 Chemin des Combes
F-06600 Antibes(FR)
Inventor: Rheinart, Charles
922 Route des Vallettes Sud
F-06140 Tourrettes/Loup(FR)

(74) Representative: Lattard, Nicole
Compagnie IBM France
Département de Propriété Intellectuelle
F-06610 La Gaude (FR)

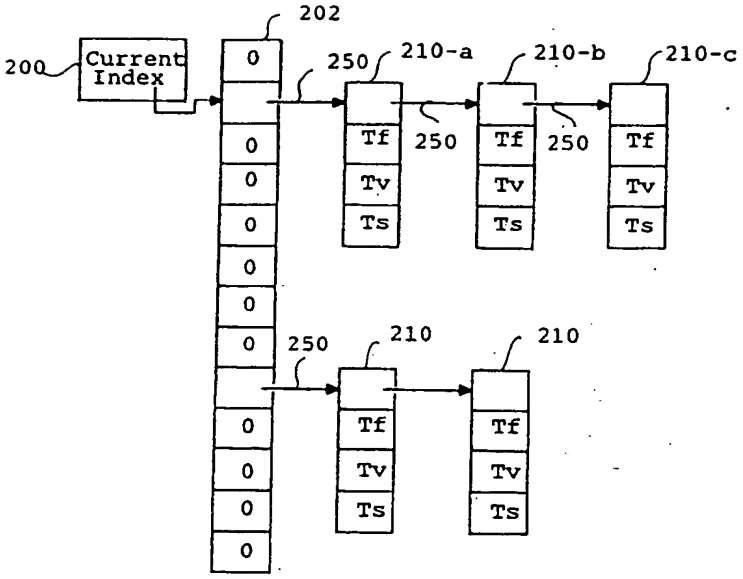
(54) Efficient multi-users timer.

(57) A system for providing a plurality of timers to perform the timing of event occurrences wherein to each event corresponds a timer control block (210) which stores in its time_flag (Tf) the indication of whether the timer control block is chained or un-chained, running or stop, in its time_out (Tv) the expiration time interval and in its time-stamp (Ts) the current time as a reference at each interruption for further treatments. The timer control blocks are chained by a one-way link (250) according to their expiration times in a way that each timer chain comprises the timer control blocks whose events will occur at the same time. A cyclic table (202) of index enables to classify the timer chains according to their expiration times, each one being pointed by an index. When a START operation is requested for an event which has to occur in a time_out value, an index is computed according to the Tv and the current time in order to insert its corresponding timer control block at the head of the timer chain pointed by said index, said timer control block storing the state of CHAINED-RUNNING in its time-flag and the current time in its time_stamp, if the timer control block is already chained, then updating only the timer_stamp to the current time and the time_flag to running because it has previously been stopped by the user. Whenever a RESTART operation is requested for an event before it has occurred, the time_stamp of the corresponding timer control block is updated to the value of the current time. Whenever a STOP operation is requested before the event has occurred, its time_flag is updated to "STOP".

Meanwhile the time_stamps and the time_flags are updated according to the START, STOP and RESTART operations, the current index of the cyclic table is incremented at each time_tick to delete the timer control blocks of the chain whose events have occurred or whose time_out values have expired, and to insert new timer control blocks in the new timer chain for those which have been interrupted and whose events have not occurred yet.

EP 0 586 768 A1

FIG . 3 Timer chain structure



Field of the invention

The present invention relates to a method and a device implemented in a data processing systems wherein a plurality of users need to watch whether events occur before time_out delays elapse, and more particularly to such a method and a device to be implemented in a communication system, wherein users exchange messages, for notifying the users that predetermined time_out delays assigned to events have elapsed.

Background art

A timer is a device which can be set to furnish an interrupt or a timeout indication, at a specific time instant or after a selected time interval. Timers are required in communication systems in which protocols need that a very large number of simultaneously occurring tasks or events be supervised to detect whether they occurred within predetermined delays. A START operation is sent by the user to start the timer in order to supervise a corresponding event. When the supervision of an event has to be interrupted for different reasons, a STOP operation is generated by the corresponding user. After a while, the supervision of the corresponding event may be requested to start once again, then a START operation is generated by the user. While the timer associated to an event is still running, the user may request a RESTART operation in order to delay the timing of the corresponding event.

In communication systems, because the transmission time of messages between users is very short, thousands of START, RESTART and STOP operations are generated at nearly the same time by the users for supervising many events. In that environment, these operations need to be performed very efficiently in order not to impair the performance of the communication systems.

Timer arrangements generally comprise a timer control block (TCB) associated to each event the time-out delay of which is controlled. These TCB are managed (i.e. chained, updated or removed from a chain under control of a program. Figure 1-A represents a simple chain structure of the TCB which are doubly linked in order to make easier the deletions and the insertions of TCBs in the TCB chains: each new TCB is simply added to the end of the chain in response to a START operation and the timer which has to be deleted in response to a STOP operation is easily removed from the chain thanks to the double links. That simple structure is not adapted to control a large number of outstanding events because at every time increment (time_tick), it is necessary to scan the entire chain to detect if the time_out delay of each event has elapsed.

This scan on each timer tick can be avoided by chaining the TCB in the order of which the corresponding events are awaited. Figure 1-B shows the implementation of an ordered list TCB chain structure whose performance is higher than the previous one's. In the subject mechanism, a new TCB is inserted in the correct point of the chain. This operation is facilitated by the fact that the TCB are doubly linked, however, it is done at the cost of introducing a chain scan every time the supervision of a new event is needed. This implementation is possible when the system does not require several thousands of timer control blocks.

The EP application A1 355 243 discloses a timer device which enables, in a multiple timer arrangement, simple setting and cancelling of timers where, to each timer correspond a timeout value and a timer tag. A specific time scale register and a clock divider are provided so that the resolution of the timer is easily selectable. To allow existence of several timers expiring at the same time, a special timer service with chainable timer control block is provided in order to delete all a chain of timers which expire at the same time. A drawback of this device is that it is not adapted to an environment where the START, RESTART and STOP operations may occur one after the other almost simultaneously before the timeout occur, which will therefore complicate the updating of the timeout which has to be added to the current contents of the cyclic counter to generate a new address at which the timer tag is stored thus establishing a new timer.

Objects of the invention

An object of the invention is to provide a method and a device for supervising a large number of events in a data processing system in a very efficient and simple way using a large number of timers.

It is another object of the invention to provide such a method and a device which associate to each timer a timer control block, and wherein all timer control blocks are managed in order to improve the START, RESTART and STOP operations of the timers.

Summary of the invention

These objects are achieved by a method of controlling a plurality of timers of different users, each timer being associated to an event and to a timer control block in a data processing system wherein a user issue a START operation of a timer when the occurrence time of the corresponding event is awaited within a time_out (Tv), a RESTART operation when the occurrence time of an

event has to be delayed, and a STOP operation when the event has occurred, said method being characterized in it that:
each timer control block is divided into at least a flag field, a time__stamp field and a time__out value field and in that it comprises the steps of:

- providing a cyclic means having a number N of storing locations which are sequentially addressed by an addressing means at regular time intervals (timer__tick),
- in response to a START operation issued by a user
  - computing, according to the current time and the time__out value of said event, an address of a storing location in said cyclic table,
  - inserting said timer control block in a chain of timer control blocks associated to events which expire at the same time, said chain being pointed by a control block address stored at the computed address in the cyclic table,
  - updating and storing in the timer control block the flag__state field of the time__flag to the state of "RUNNING" to indicate that the timer is active and the flag__chain field of the time-flag to the state of "CHAINED" if the timer control block is inserted for the first time in the chain, and if the timer control clock is already chained then the time__state is simply updated to the state of "RUNNING",
  - storing the time__out value in the timer control block of the corresponding event,
  - updating and storing in the timer control block the time__stamp to the current time,
- in response to a STOP operation updating the flag__state in the time__flag field of the associated timer control block to the state of "STOP",
- at each regular time interval (timer__tick), successively reading each timer control block chained to the addressed location of the cyclic means, and checking the state of the time__flag field and
  - unchaining the timer control block if its flag__state is STOP
  - otherwise, computing the new time__out according to the current time, the time__stamp and the last time__out (new time__out = time__out + time__stamp - time__current) to control if the time__out delay has elapsed and inserting the timer control block to a new timer control block chain if the value of the new time__out is positive, or unchaining and stopping the timer control block if said timer control block is equal to 0 and notifying the user that the time__out delay has elapsed.

Said method is implemented in a device which comprises:

- a cyclic means having a number N of storing locations which are sequentially addressed by an addressing means at regular time intervals (timer__tick) in order to classify the different timer control blocks of the corresponding events according to their time__out values and the current time, and
- a timer chain which is pointed by said addressing means and to which are chained a plurality of timer control blocks associated to events which should occur at the same time before said START, RESTART or STOP operations interrupt the timing of said events.

Brief description of the drawings

Figures 1-A and 1-B represent the evolution of the timers in the prior art, from a simple timer chain structure to an ordered list timer chain structure.

Figure 2 shows the environment of the preferred implementation of the present invention.

Figure 3 shows the timer chain structure of the present invention.

Figure 4-A shows an example of a RESTART operation flow.

Figure 4-B shows an example of STOP operations followed by START operations flow.

Figure 5 shows the flow-chart of the INIT operation.

Figure 6 shows the flow-chart of the START operation.

Figure 7 shows the flow-chart of the TIMER__TICK operation.

Detailed description of the present invention

Figure 2 shows an environment wherein the present invention may be implemented. It comprises a switching system (100) which enables to connect a plurality of users (106). A processor (102) controls the exchange of messaged between the users and according to the subject invention manages the timer arrangement. A memory (104) which contains the communication control program and also the data structure of the timer control blocks. Each user has its own timer control block in the memory, it is identified by an address. The timer control program which enables to manage the timer control blocks according to the START, RESTART and STOP operations is also stored in the memory and is controlled by the processor.

In order to understand figure 3, one must have in mind the three following notions:

- The current time indicated the value of the incrementation of the timer tick. This variable is reset to 0 at the initialization time.
- The current index indicates the position of the index in the cyclic table corresponding to the current time. It is incremented at each timer tick. When the index reaches the end of the table, it is reset to 0.
- The current chain indicates the sets of timer control blocks specified by the current index.

Figure 3 shows the timer chain structure of the present invention. It comprises a cyclic table (202) having a given number of storing locations addressed by a unique current index (200). This current index is incremented at regular time interval called timer_tick to specify the corresponding storing location which store the address of a timer control block to be chained to the cyclic table. Timer control blocks (210-a, 210-b, 210-c) may be chained in one-way link (250) to the cyclic table as will be described later on, by storing the address of the next timer control block in the previous TCB if there is one, otherwise to use a special code, for example 0, to indicate that it is the last block of the timer chain. The current index is used when adding a TCB to the timer chains -the chain number or the index of the cyclic table to which the timer block has to be added is relative to the current chain, not to the first chain. On each timer tick, the current index is incremented, and the chain specified by the new index is now the current chain.

In addition to the conventional address fields in the TCB used for the chaining purpose they comprises specific fields, as follows:

DATA STRUCTURE:

- The time-flags field (Tf) indicates the status of a timer control block associated to an event. It comprises two time flags:
  - The flag_chain: having two values CHAINED or UNCHAINED to indicate whether the timer control block is in a chain or not,
  - The flag_state: having two values STOP or RUNNING to indicate whether the timer is active or not.
- The time_value field (Tv) indicates the time_out values of the associated event. Its range varies from: 1 second to 10 seconds. By time_out it is meant the time duration after which a user should be notified that an awaited event has not occurred.
- The time-stamp field (Ts) indicates when the last START or RESTART operation is performed.

The three values Tf, Tv and Ts are stored in the timer control blocks.

To each timer control block corresponds an event which requires three kinds of information which are: the time_flags, the time_value and the time_stamp whose values are updated according to the START, RESTART or STOP operations.

There two categories of operations which are performed for the management of the timer control blocks: the operations of START, RESTART and STOP that are requested by the users in order to chain, unchain or update the timer control blocks, and the operation which is performed at regular time interval (at every timer_tick) which enables to check if the events whose timer control blocks are chained to that timer chain have occurred or not in order to cancel the whole chain and to chain the new timer control blocks whose events have not occurred to a new timer chain.

START, STOP AND RESTART OPERATIONS:

The START operation is used in the present implementation to add the timer control block associated to an event into a timer chain whose events are initially assumed to occur before the same time. It is performed as follows:

- If the timer control block is not chained, it is added to a new timer chain. Its time_flag field (flag_chain and flag_state) is updated to the state of CHAINED_RUNNING and its time_stamp field (Ts) is updated with the current time.
- If the timer control block is already chained, then its flag_chain is updated to the state of RUNNING and its time_stamp field Ts is updated with the current time.

The STOP operation does not remove at once the timer control block from the chain. It does only specify the state of the timer. The timer control blocks will be removed once all the events have already occurred and once the timer control block, which has been updated according to the interruptions of the process while it was running, has expired. This allows unnecessary removal and addition of timer control blocks from /into the chains and improve the timer control block management.

The STOP operation simply updates the flag_state of the specified timer control block to STOP and keeps it on the timer chain.

The RESTART operation does not remove the associated timer control block from the timer chain and set up a new timer event by chaining it on a new chain, but it does only update the timer control block according to the current time as long as the RESTART operation occurs before the STOP operation.

The RESTART operation simply updates the time_stamp to the current time.

FAST RESTART OPERATION

As it is useless to unchain the timer control block and chain it again when the RESTART operation is invoked before the current index reaches the position of the chain number to which said timer control block is chained, several fast RESTART operations for the duration of the time_out value. The present implementation requires only one manipulation of the timer control block for the duration of the time_out value, comparing to one manipulation at each RESTART operation in other implementations.

It must be acknowledged that this implementation is particularly adapted to an environment where the users exchange information frames and where the transmission time of each frame is around 1 micro-second, and where a number of 1000 frames are to be sent per second, this leads to 1000 restart operations.

It should be noticed that a START operation on an already active timer is implemented as an implicit RESTART operation.

The timer flags are widely used in the START, STOP and RESTART to handle correctly the state of the timer and to reduce its manipulation (add to the timer chain, remove from the timer chain). When a timer control block is not chained, this implies the timer is non-active. But a nonactive timer control block does not imply that the timer control block is unchained. Therefore, the relation ship among the states CHAINED/UNCHAINED and STOP/RUNNING are as follows:

UNCHAINED implies STOP; RUNNING implies CHAINED.

And the correct combinations of these states in a timer control block are: UNCHAINED-STOP, CHAINED-STOP and CHAINED-RUNNING.

OPERATION RELATED TO THE CURRENT CHAIN (TIMER_TICK)

One TCB chain which is the current chain, is processed at each timer tick, and the chain becomes empty after this process. All TCBs on the current chain are processed as follows:

- The timer control blocks which are marked STOP are simply removed from the timer chain.
- The timer control blocks which are marked RUNNING and where the condition (current_time < time_stamp + time_value) is true are added to a new chain. The index of the new chain is specified by the remaining duration: (time_stamp + time_value - current time).
- The timer control blocks which are marked RUNNING and where the above condition condition described is false, are removed from the chain and the users supervising the corresponding events are notified that the time_out delays of the events have expired.

HIGH CAPACITY AND EFFICIENT TIMER SUPPORT:

Each active timer has a time_out value which is translated by the timer control program to an index associated to the cyclic table (202). The index gives the chain number to which the timer control block has to be inserted. Each new timer control block is simply added to the head of the chain. This insertion is easy to perform thanks to the one-way link of the chain. Since the timer control block is always inserted to the head of the chain and since all the timer control blocks of a chain are deleted at the same time, it is not necessary to implement a doubly link structure which operates slower than a one-way link.

It can be seen that no scan is required for any of these timer control blocks, neither for addition nor for cancellation of a timer control block, nor when the timer tick occurs. The cost of adding, cancelling and notifying the user is approximatively constant, regardless of the number of outstanding timer events. The number of timers is configurated according to the memory space allocated to the timer control blocks.

Figure 4-A shows an example of a RESTART operation flow between a sender (301) connected to its own timer and a receiver (302) also connected to its own timer. But in this example, we will only consider what happens on the sender side.

A sender (401) issued a START operation at T0 with a time_out value (Tv) when it sends a data (0) of a message to a receiver (402). The time_flags and the time_stamp are updated respectively to CHAINED-RUNNING and to the current time, Ts = T0. The associated TCB is added in the chain of TCBs which correspond to events which will occur before time T, where T = T0 + Tv. At T1, the receiver sends a partial acknowledgement (2) for the reception of the data (0) and the data (1) whereas the sender has already sent the data (2), then the timer is restarted, not stopped because the data (2) have not been acknowledged by the receiver. As it is a RESTART operation, the time_stamp is updated with the current time, Ts is set equal to T1 in the control block.

In this example, the timer is restarted each time the sender receives an acknowledgement because the last data sent have not been acknowledged yet, and the timer_stamp is updated with the current time which is successively T2 and T3.

When the current time is equal to T (at time T), the timer control block is removed from the TCB chain and is added in the chain of TCBs corresponding to events will occur at time T' = Tv + T3, and the time_stamp is left unchanged Ts = T3.

EFFICIENT START/STOP OPERATION

When the sender issues a STOP operation, the timer control block is kept on the chain. This provides an opportunity to support efficiently multiple START and STOP sequences as is illustrated in figure 4.

Figure 4-B shows an example of STOP operations followed by START operations between a sender (401) and a receiver (402) which are combined with RESTART operations.

In response to the START operation received by the processor, the timer control program code starts a the supervision of the timer control block corresponding to the event at T0 with a time_value (Tv) then the user sends a data (0) of a message. The time_flag and the time_stamp are updated respectively to CHAINED-RUNNING and to the current time, Ts = T0. The associated timer is added in the chain of TCB associated to events which are assumed to occur before the time T, where T = T0 + Tv. At T1, the user receives a complete acknowledgement (2) for the reception of the data (0) and the data (1), it sends a STOP operation and the flag_state is updated to STOP. Another message may be required to be sent out at T2 and the user has to send a START operation again, therefore, the flag_state and the time_stamp are respectively updated to RUNNING and to the current time Ts = T2.

At time T5, when the user receives a partial acknowledgement, then a RESTART operation is sent as it has been seen in the example of figure 3.

When the current time is equal to T (at time T), the timer control block having the expiration time T is simply removed from the timer chain.

In this example where partial and complete acknowledgements interleave, the sequences of START, STOP and RESTART operations at time T1 through T6, do not involve the manipulation of the timer control blocks because the time T has not been reached yet, (T = T0 + Tv). This opportunity enables to gain performance as long as the START, STOP and RESTART operations occur almost simultaneously.

The present implementation enables to reduce the manipulation of the timer control blocks by delaying the effective deletion of the TCBs which remain chained to the timer chain until the time_out delay has elapsed. Again, this implies only one manipulation of the timer control block, comparing to one adding and removing the timer control block to/ from the timer chain at each START and STOP operation.

Figure 5 shows the flow-chart of the initialization operation of the timer control blocks.

The current_time which is an integer to be incremented at each timer_tick along the processing is initially reset, at step 501, so does the current_index in order to point to the first storing location of the cyclic table, at step 502. A loop is entered to scan every location of the cyclic table to set the table(index) in each location to 0. The index which is also an integer varying from 0, at step 503, to max_entry, at step 504, enables to classify the storing locations of the cyclic table. The max_entry number corresponds to the size of the cyclic table which may be changed in order to enable to take into account the events whose time_out value are great. Each table(index) corresponding to a storing location is set to 0 in order to indicate that there is no timer control block addressed by said storing location of the cyclic table, at step 505. Then the index is incremented by 1, at step 506.

A table(index) is set to 1 to indicate that a timer control block is chained, and the corresponding storing location will store the address of said timer control block in its memory.

Figure 6 shows the flow-chart of the START operation of a timer control block.

At step 601 it is tested if the flag-state is in the state of stop (=0) or running (=1). If the timer control block is not active (flag-state = 0), then the flag-chain is tested to check if the TCB is chained to the cyclic table, at step 602. If it is not chained (flag-chain = 0), then the value of the index which represents the remainder of the division of (time_out value + current index) by max_entry is calculated at step 603. At step 604, the TCB is chained to the cyclic table at the position which corresponds to the calculated value of the index by a one-way link. The timer control block is therefore inserted at the head of the corresponding timer chain, at step 605. At step 606, the time_stamp of the timer control block is updated to the value of the current time, before the start operation ends.

If at step 601, the flag-state is found running then the start operation ends.

If at step 602, the timer control block is chained (flag-chain = 1) and stopped as indicated by the test at step 601, the program will loop to step 606 to update the time_stamp to the value of the current time.

A START operation on an already active timer is implemented as an implicit RESTART operation.

The STOP operation consists only in setting the flag_state to STOP and the flag_chain to CHAINED.

The RESTART operation consists only in updating the time__stamp to the current time.

Figure 7 shows the flow__chart of the TIMER__TICK operation performed at regular time interval (timer__tick).

At every timer__tick, the current__time is incremented, at step 701, and the current__index is computed as the remainder of the division of (current__index + 1) by max__entry, at step 702. The system checks the timer control blocks which are chained to the cyclic table at the corresponding current__index, at step 703.

At step 710, the first timer control block of the chain is pointed. If there is a timer control block which attached to the corresponding current__index, at step 720, the system checks if the timer control block is running or stop according to the state of the flag__state. If it is running, then at step 721, the remain__time is computed (as remain__time = time__stamp + time__out value - current__time).

If the remain__time is equal to zero, at step 730, which means that the time__out delay corresponding to the timer control block has elapsed. Then, at step 731, the flag__state is set to stop, and the flag-chain is also set to unchained, at step 732. The system notifies the corresponding user that the time__out delay has elapsed so that he may take the appropriate recovery procedure, at step 733.

If the remain__time is not equal to zero, at step 730, which means that the event has not occurred yet, then the system computes the new index which has been modified by the value of the time__stamp (index = the remainder of the division of (remain__time + current__index) by max__entry), at step 734. At step 735, the TCB is inserted at the head of the chain which corresponds to the number of the new index. The timer control block stores therefore the new values of the time__flag, time__value, and time__stamp, at step 736.

If, at step 720, the flag__state of the present timer control block indicates that the TCB is stopped, then the system unchains the timer control block from the chain.

Once the first timer control block is checked, after steps 733 or 736 or 722, then the system points to the next timer control block at step 740. The system loops back to step 710 and the processing starts once again if there are still timer control block on the same chain.

Otherwise, at step 750, the system resets the whole chain of timers by cancelling all the timer control blocks of that chain.

Then, the system waits until the next time__' ': in order to treat in the same way the timer control blocks as it is previously described. Meanwhile, all new events to which are associated timer control blocks are inserted at the head of the timer chain which corresponds to the index computed according to the time__value and the current__index.

Claims

1. A method of controlling a plurality of timers of different users (106), each timer being associated to an event and to a timer control block (210) in a data processing system wherein a user issues a START operation of a timer when the occurrence time of the corresponding event is awaited within a time__out (Tv), a RESTART operation when the occurrence time of an event has to be delayed, and a STOP operation when the event has occurred, said method being characterized in that:
each timer control block (210) is divided into at least a time__flag field (Tf), a time__stamp field (Ts), a time__out value field (Tv) and an address field, and
in that it comprises the steps of:
   - providing a cyclic means (202) having a number N of storing locations which are sequentially addressed by an addressing means (200) at regular time intervals (timer__tick),
   - in response to a START operation issued by a user (106)
     - computing, according to the current time and the time__out value of said event, an address of a storing location in said cyclic means (202),
     - inserting said timer control block (210-a) in a chain of timer control blocks (250) associated to events which expire at the same time, said chain being pointed by a control block address stored at the computed address in the cyclic means,
     - updating the flag__state field of the time__flag (Tf) to the state of "RUNNING" to indicate that the timer is active and the flag__chain field of the time__flag to the state of "CHAINED" if the timer control block is inserted for the first time in the chain, and if the timer control clock is already chained then the flag__state is simply updated to the state of "RUNNING",
     - storing the time__out value (Tv) in the timer control block of the corresponding event, and
     - updating the time__stamp (Ts) to the current time,
   - in response to a STOP operation updating the flag__state in the time__flag field

of the associated timer control block to the state of "STOP",

- at each regular time interval (timer_tick), successively reading each timer control block (210) chained to the storing location of the cyclic means (202), and checking the state of the time_flag field (Tf) and
  - unchaining the timer control block if its flag_state is STOP,
  - otherwise, computing the new time_out value (Tv) according to the current time, the time_stamp (Ts) and the last time_out (new time_out = time_out + time_stamp - current time) to control if the time_out delay has elapsed and inserting the timer control block to a new timer control block chain if the value of the new time_out is positive, or unchaining and stopping the timer control block if said new time_out is equal to 0 and notifying the user that the time_out delay has elapsed.

2. The method of claim 1 further comprising the step of:
   - in response to a RESTART operation issued by a user when the occurrence of an event has to be delayed, updating the time_stamp of the associated timer control block to the current time.

3. The method of claims 1 or 2 wherein the timer control blocks are chained to each other by a one-way link (250).

4. A device for controlling a plurality of timers of different users (106), each timer being associated to an event and to a timer control block (210) in a data processing system wherein a user issues a START operation of a timer when the occurrence time of the corresponding event is awaited within a time_out (Tv), a RESTART operation when the occurrence time of an event has to be delayed, and a STOP operation when the event has occurred, said device being characterized in that:
   each timer control block (210) is divided into at least:
   - a time_flag (Tf) which includes a flag_state set to the state of "RUNNING" at each START operation to indicate that the event is awaited or set to the state of "STOP" at each STOP operation to indicate that it has already occurred and a flag_chain to indicate if the timer control block is "CHAINED" or "UNCHAINED" to a timer chain (250),
   - a time_stamp field (Ts) which is set to the current time at each START operation and updated to the current time at each RESTART operation to enable the event to be delayed,
   - a time_out value field (Tv), and
   - an address field,

   and characterized that it comprises:
   - a cyclic means having a number N (max_entry) of storing locations in order to classify the different timer control blocks of the corresponding events according to their time_out values and the current time,
   - an addressing means (200) which sequentially addresses a storing location of said cyclic means at regular time intervals (timer_tick), and
   - a timer chain (250) pointed by said addressing means (200) and comprising a plurality of timer control blocks associated to events which should occur before the same time T, before said START, RESTART or STOP operations interrupt the timing of said events, said timer control blocks being chained to each other by storing in the address field the address of the next timer control block or a special code to indicate the end of the chain,

   wherein at regular time interval (timer_tick), each time control block chained to the addressed location of the cyclic means (202) is read:
   - to unchain the time control block if its flag_state is STOP,
   - otherwise, to compute the new time_out value (Tv) according to the current time, the time_stamp (Ts) and the last time_out (new time_out = time_out + time_stamp - current time) to control if the time_out delay has elapsed and to insert the timer control block to a new timer chain if the value of the time_out is positive, or to unchain and stop the timer control block if said timer control block is equal to 0 and to notify the user that the time_out delay has elapsed.

5. The device according to claim 4 wherein the timer control blocks (210) are chained to each other by a one-way link (250).

FIG. 1 Timer chain structure in the prior art

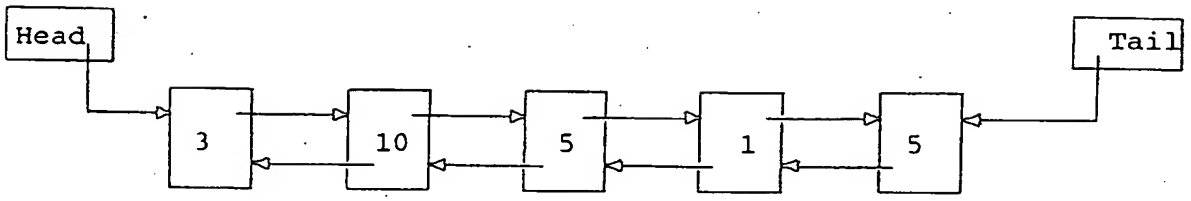


FIG. 1-A Simple timer chain structure

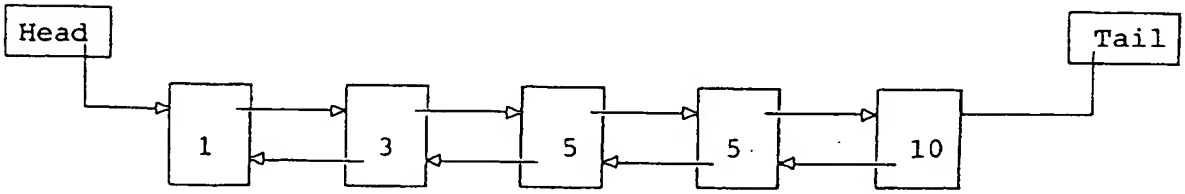


FIG. 1-B Ordered list timer chain structure

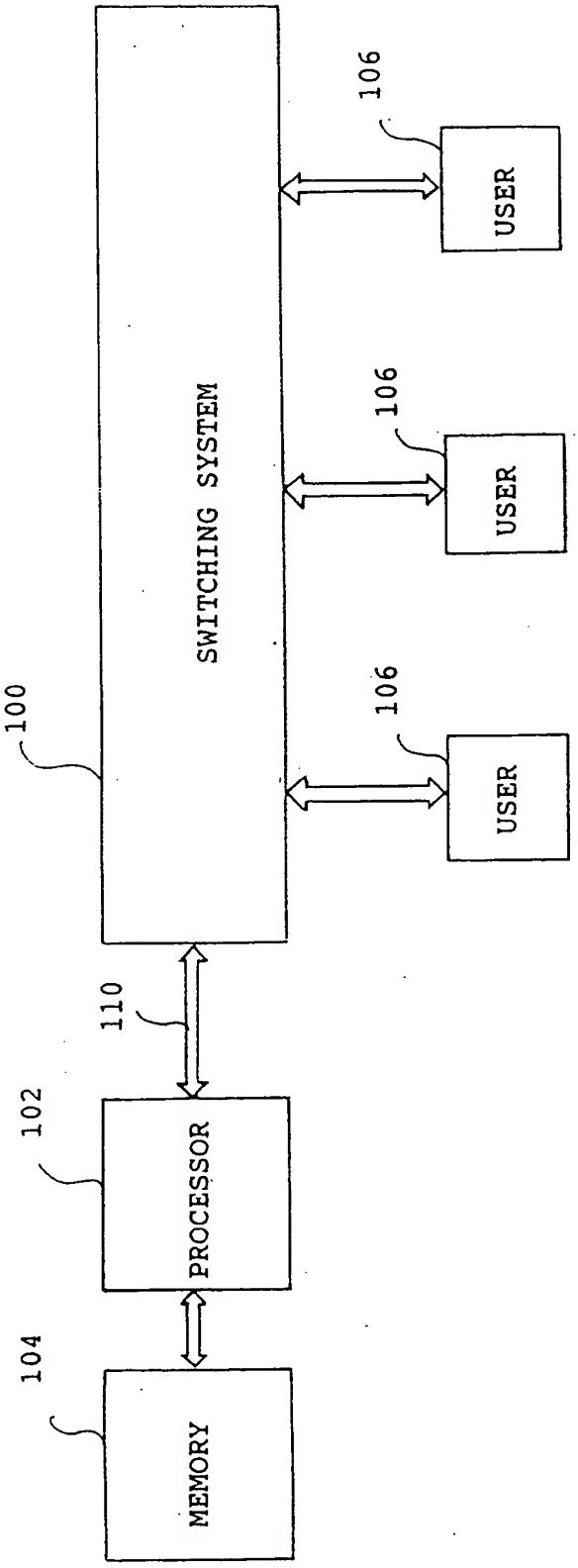


FIG. 2 ENVIRONMENT OF THE IMPLEMENTATION OF THE PRESENT INVENTION

FIG. 3 Timer chain structure

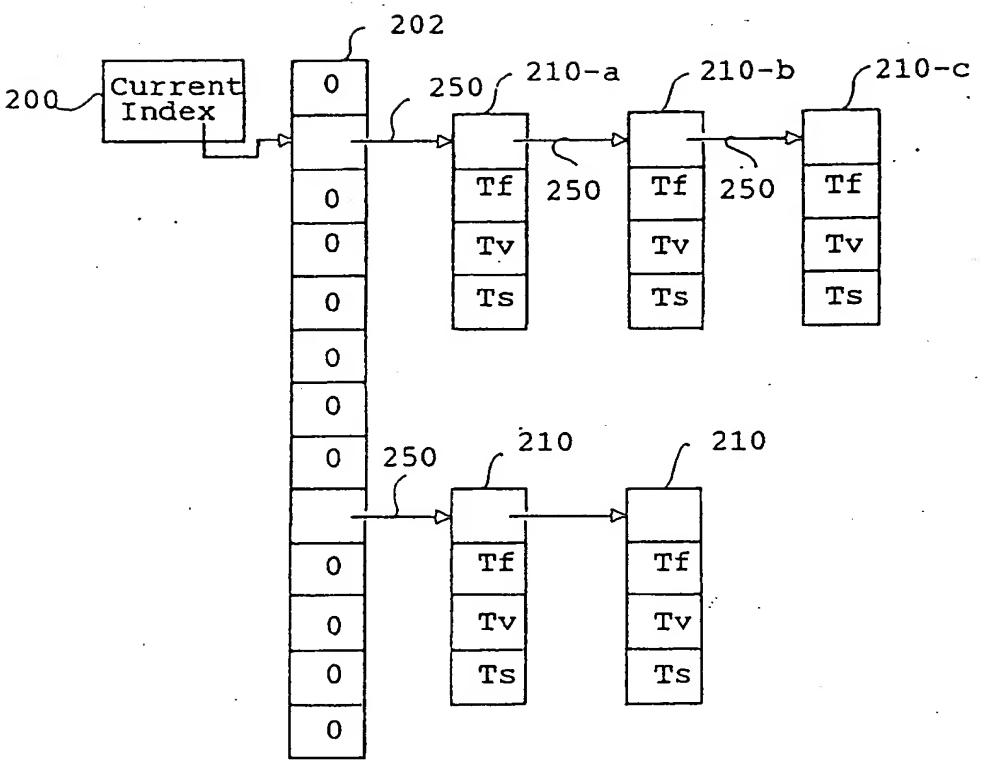


FIG. 4-A Example of a restart operation flow

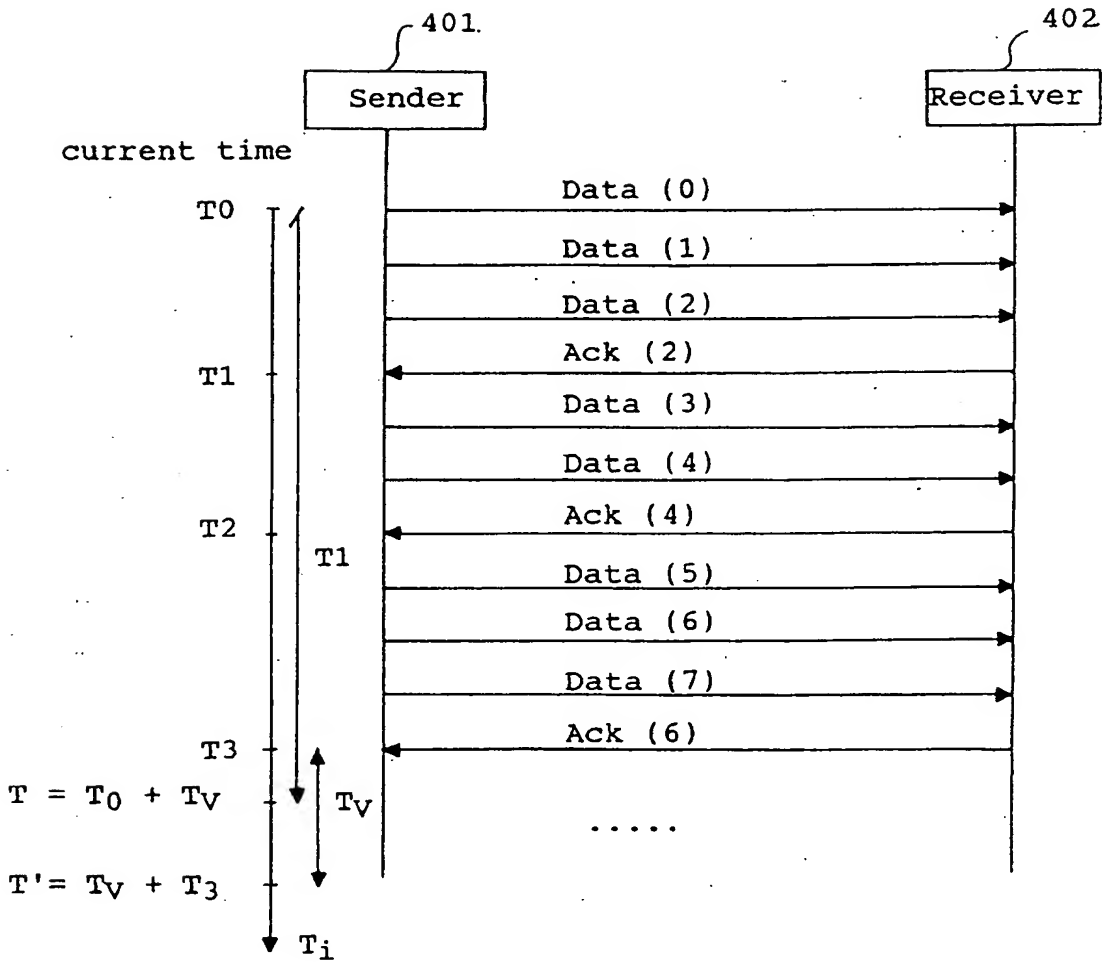


FIG. 4-B Example at STOP operations followed by START operations flow

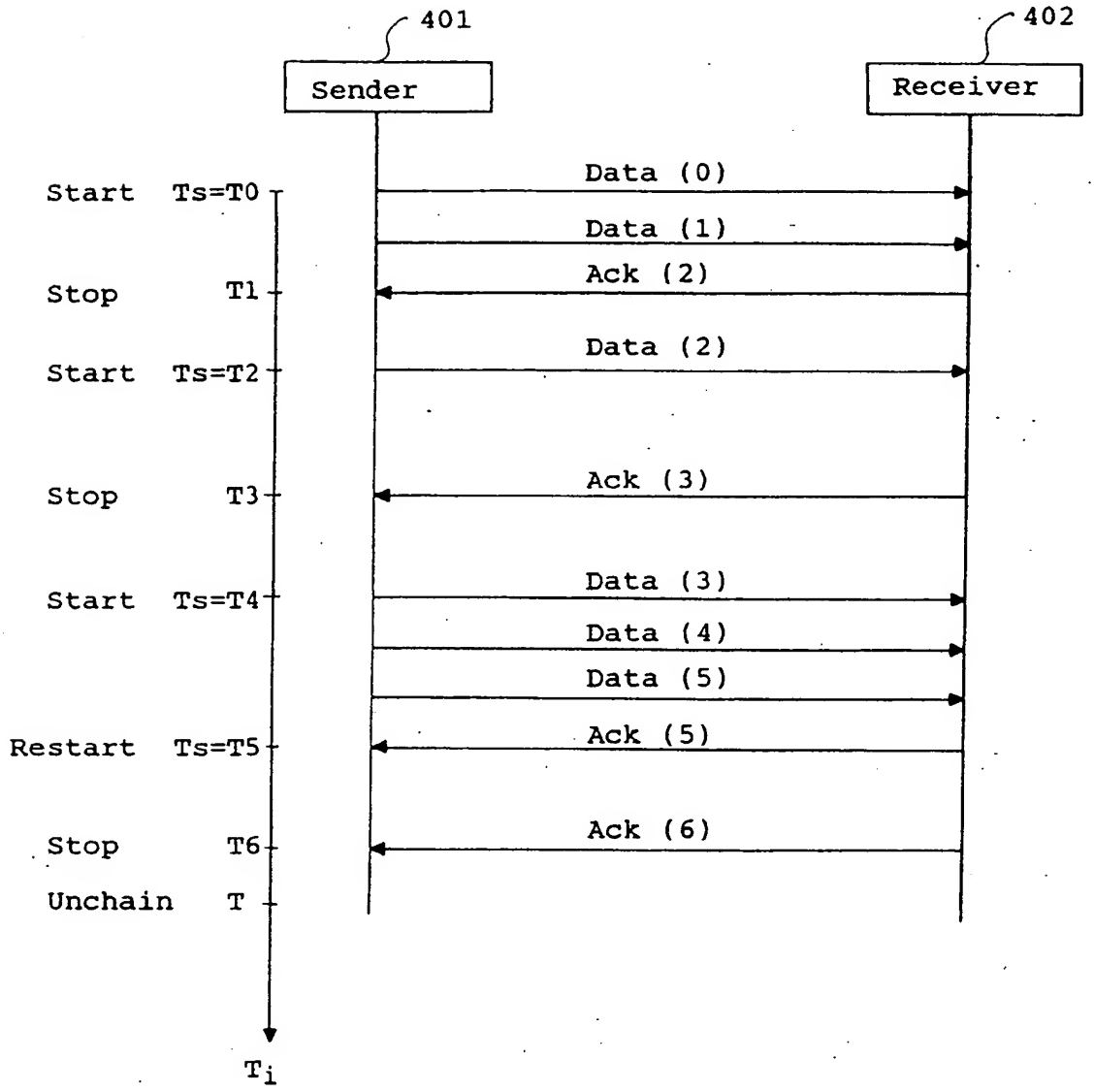


FIG. 5 INIT operation

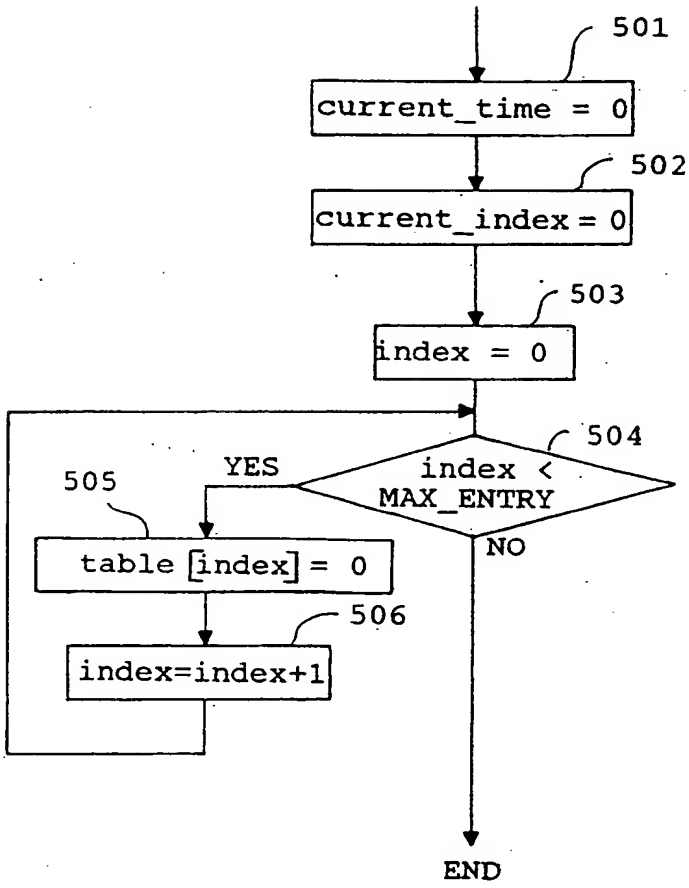


FIG. 6 START operation

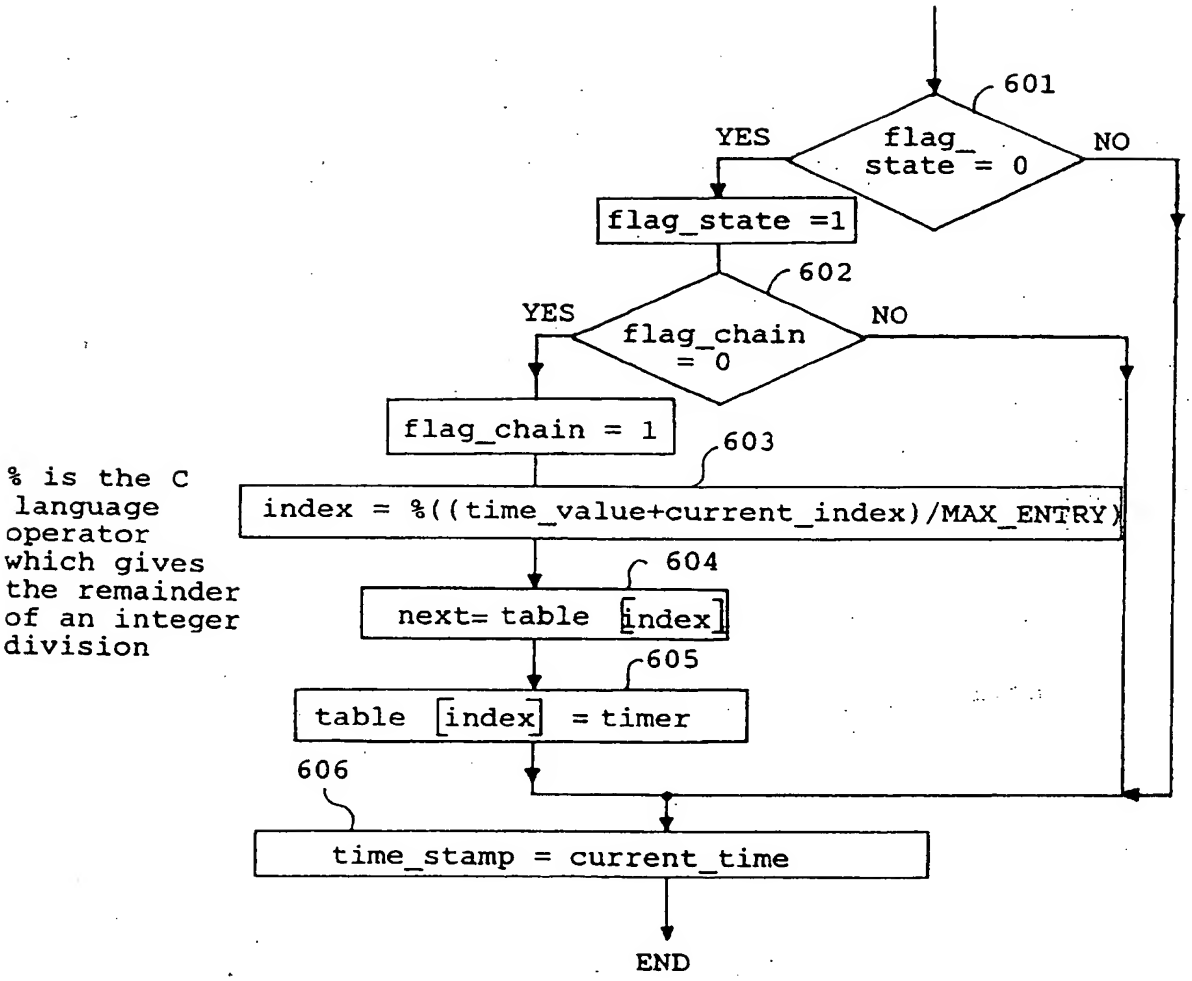
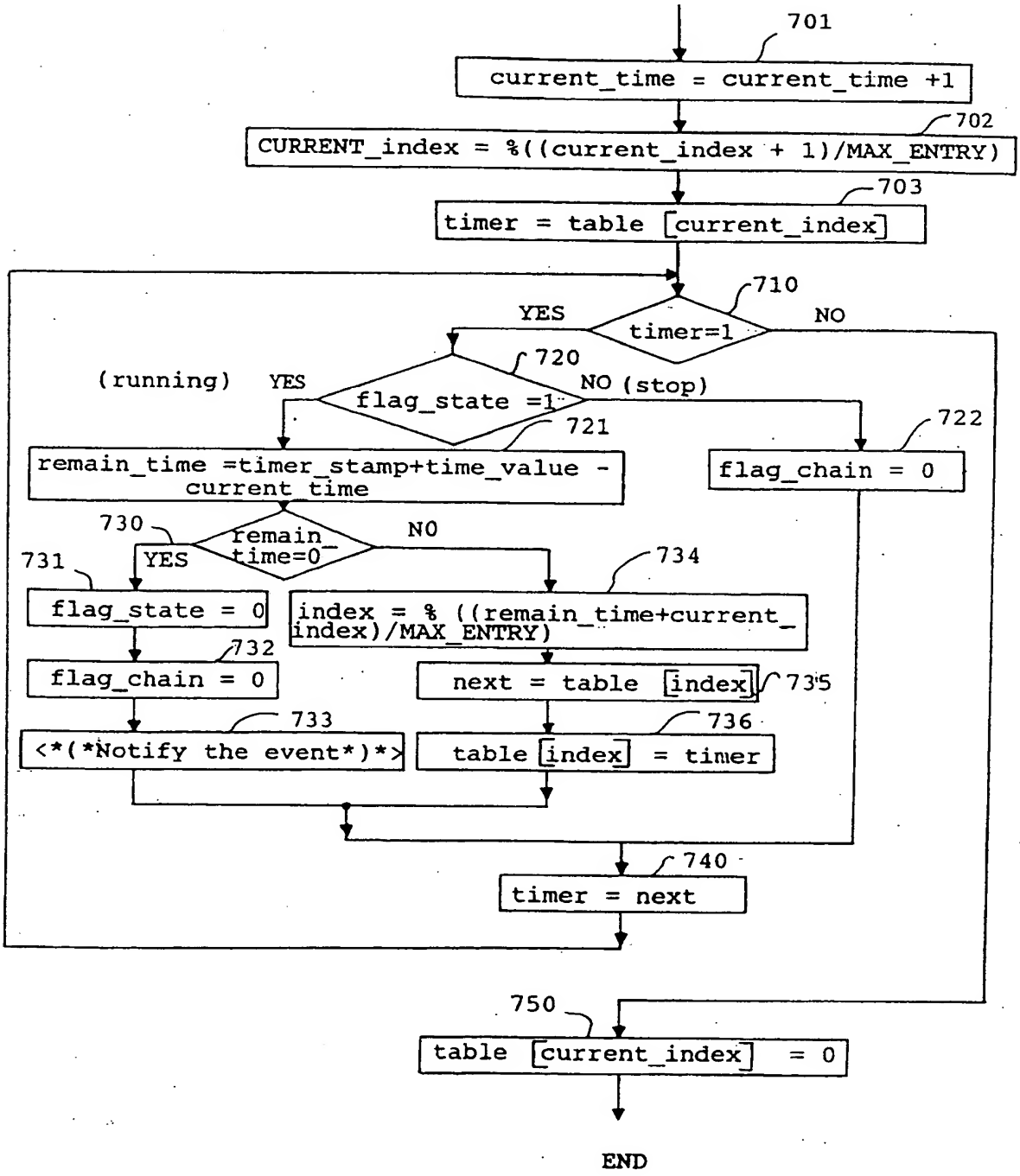


FIG. 7 TIMER_TICK operation



European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 92 48 0130

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
D,A	EP-A-0 355 243 (IBM CORPORATION) * abstract * * page 3, column 4, line 31 - line 48 * * page 4, column 6, line 10 - line 47 * * page 7, column 11, line 15 - column 12, line 37; figure 5 *	1,3-5	G06F9/46 H04L29/06
A	IBM TECHNICAL DISCLOSURE BULLETIN vol. 33, no. 12, May 1991, ARMONK, NY, US pages 206 - 207 'Binary-Weighted Delay Function' * the whole document *	1,3-5	
A	IBM TECHNICAL DISCLOSURE BULLETIN vol. 32, no. 6B, November 1989, ARMONK, NY, US pages 266 - 270 'Mechanism for a wraparound time-ordered list with a variable real-time list origin' * page 266, line 1 - line 9 * * page 267, line 18 - page 268, line 4; figures 1-3 * * page 270, line 16 - line 40 *	1,3-5	TECHNICAL FIELDS SEARCHED (Int. Cl.5) H04L G06F
A	PROCEEDINGS OF THE 1988 INTERNATIONAL ZURICH SEMINAR ON DIGITAL COMMUNICATIONS 8 March 1988, pages 93 - 98 E. MUMPRECHT ET AL. 'Timers in OSI protocols - Specification versus Implementation' * page 93, right column, paragraph 1.2 * * page 97, left column, paragraph 5 - page 98, left column, paragraph 5.2 *	1,3-5	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	01 JUNE 1993	BRAVO P.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0401)